

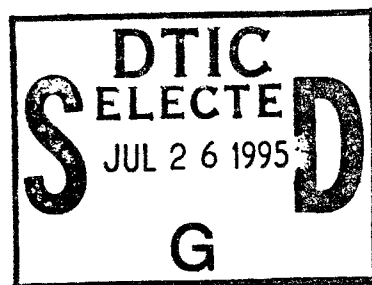
RL-TR-95-98
Final Technical Report
May 1995



APPLICATIONS AND ENABLING TECHNOLOGY FOR NYNET UPSTATE CORRIDOR

Syracuse University

Salim Hariri and Geoffrey Fox



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

19950724 184

DTIC QUALITY INSPECTED 5

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-95-98 has been reviewed and is approved for publication.

APPROVED: *Richard C. Butler II*

RICHARD C. BUTLER II
Project Engineer

FOR THE COMMANDER:



JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3BC) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE June 1995		3. REPORT TYPE AND DATES COVERED Final Aug 93 - Sep 94	
4. TITLE AND SUBTITLE APPLICATIONS AND ENABLING TECHNOLOGY FOR NYNET UPSTATE CORRIDOR				5. FUNDING NUMBERS C - F30602-93-C-0195 PE - 62702F PR - 4519 TA - 22 WU - PJ	
6. AUTHOR(S) Salim Hariri and Geoffrey Fox					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Northeast Parallel Architectures Center (NPAC) Syracuse University Syracuse NY 13244-4100				8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (C3BC) 525 Brooks Rd Griffiss AFB NY 13441-4505				10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-95-98	
11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: Richard C. Butler II/C3BC/(315) 330-7751					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This work demonstrates the new possibilities in high performance computing, enabled by recent advances in high speed networks. This report documents the results of running a number of applications over NYNET, central New York's high speed network.					
14. SUBJECT TERMS NYNET, High performance networks, High Performance Computation and Communication (HPCC), Supercomputing				15. NUMBER OF PAGES 56	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL		

Contents

1	Introduction	5
2	NYNET High Performance Computation and Communication (HPCC) Applications	7
2.1	Multi Target Tracker	7
2.1.1	Concurrent Multi Target Tracking (CMTT)	7
2.1.2	Performance Results	10
2.2	Financial Modeling Application	12
2.2.1	Introduction and Problem Description	12
2.2.2	System Configuration and Integration	13
2.2.3	Performance Analysis	17
2.2.4	Conclusion	19
2.3	Electromagnetic Scattering	19
2.3.1	Introduction and Problem Description	19
2.3.2	System Configuration and Integration	23
2.3.3	Performance Analysis	25
2.3.4	Conclusion	25
2.4	Parallel JPEG	25
2.4.1	Problem Description	25
2.4.2	System Environment and Configuration	26
2.4.3	Implementation Description	27
2.4.4	Performance Results	27
2.5	Syracuse Language Systems	28
2.5.1	Problem Description	28
2.5.2	System Environment and Integration	29
2.5.3	Performance Issues	30
2.6	Fractal Imaging	31
2.6.1	Problem Description	31
2.6.2	System Environment and Configuration	31
2.6.3	Implementation Description	31
2.6.4	Performance Results	32
3	Evaluations of NYNET Enabling Technology	32
3.1	Mosaic Server	32

3.2	Communique Software	35
3.2.1	Discussion	36
3.3	Benchmarking ATM and Different Platforms	36
3.4	Parallel/Distributed Software Tool Evaluation	39
3.4.1	Primitives Supported by Different Software Tools	39
3.4.2	Applications Benchmark Suite	39
3.4.3	Experimental Results	41
3.4.4	Discussion	43
4	Demonstrations	43
5	Summary and Conclusions	45

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

Abstract

Current advances in telecommunication and computing will have significant impact on the proliferation of high performance computing and communication (HPCC) applications. With these emerging technologies, it is feasible to run parallel and distributed applications across a high speed wide area network which was not possible a few years ago; the high latency and low bandwidth were the main bottlenecks for the wide area network-based computing. This has lead to the deployment of several high speed networks across the country (eg. NYNET). In this report, we describe some of the HPCC applications and our experiences and lessons learned from running them over the NYNET testbed. NYNET is one of the first wide area networks to use commercially available ATM switches and first to have an aggressive research plan to develop a wide range of large scale HPCC applications. The NYNET testbed covers all of New York State and part of Massachussets, and provides an interconnection between leading educational institutions, government laboratories and industrial labs.

The main objectives of this project were to develop and demonstrate HPCC applications and evaluate current HPCC enabling technologies. We show the benefits that can be achieved from applying HPCC technologies to implement applications encountered in military (eg. multi-target tracker), industry (eg. financial modeling), scientific applications (eg. Electromagnetic scattering) and health care. Furthermore, we benchmark and evaluate several parallel and distributed platforms and software tools for developing such HPCC applications on NYNET.

1 Introduction

The 1980s spawned a revolution in the world of computing; a move away from central mainframe-based computing to distributed networks of workstations. Today, workstation servers are fast achieving the levels of CPU performance, memory capacity, and I/O bandwidth once available only in mainframes, at a cost orders of magnitude below that of a mainframe. Workstations are being used to solve computationally intensive problems in science and engineering that once belonged exclusively to the domain of supercomputers. The 1990s will be the decade of high performance distributed computing where application programs run transparently on a collection of computers that range from supercomputers or massively parallel computers down to high performance desktop or laptop computers. Such a collection of computers and supporting software environment is called a high performance distributed system (HPDS). A HPDS gives the perception of using a single, integrated computing system where users can uniformly access and name local or remote resources, and run processes from anywhere in the system, without being aware of which computers their processes are running on.

The main objectives of this project were to develop and demonstrate HPCC applications and evaluate current HPCC technologies that can transform NYNET into a HPDS. NYNET (see Figure 1) is an ATM wide area network that covers all of New York State and part of Massachusetts. NYNET provides interconnection between many of New York State's leading educational institutions (Syracuse, Cornell, Columbia, SUNY Stonybrook, Polytechnic Institute of New York), government labs (Rome Laboratory, Brookhaven National Labs), industrial labs (NYNEX, GTE etc) and several medical institutions in New York State. Most of the wide area portion of the NYNET operates at a speed of OC 48 (2.4 Giga bits per second), while each site is connected with two OC 3 links (155 Million bits per second). We developed and ported several large scale applications (Financial Modeling, Multi-Target Tracker, Electro-Magnetic Applications etc) over NYNET and evaluated their performance.

The organization of this report is as follows. In Section 2, we describe the applications which we developed and discuss their performance over NYNET. In section 3, we evaluate the current enabling technology for developing such large scale HPCC applications. Section 4 describes some of the demonstrations given by NPAC and Rome Laboratory researchers involved in this project. Finally, we summarize the report and conclude with a discussion on future research activities on the NYNET.

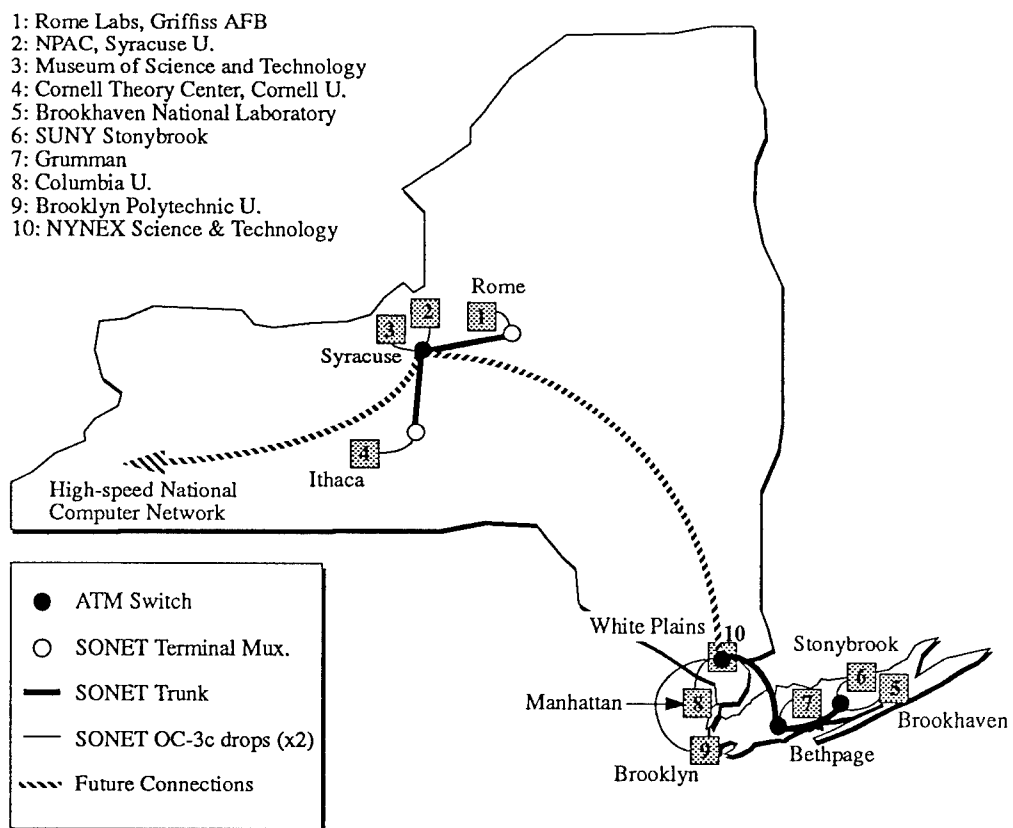


Figure 1: The NYNET ATM Testbed.

2 NYNET High Performance Computation and Communication (HPCC) Applications

2.1 Multi Target Tracker

In a previous project sponsored by Rome Labs, we modified the implementation of the tracker so that it can be easily ported using existing parallel and distributed software tools. However, in this project, we developed different parallel implementations of the tracker which are suitable for NYNET and evaluated their performance on NYNET.

The tracker demonstrates the multi target tracking capabilities that is required by a Battle Management Command, Control and Communication System. It uses an extended 3 stage Kalman filtering formalism which is the primary "tool" used to provide and sort realistic data. This filtering formalism is general and can be used in problems related to pattern recognition, signal and image processing. The 3 stage filter model has helped the development of a concurrent version of the tracker [1].

The multi target tracker is designed to provide an estimation of launch vehicle parameters for individual targets/missiles in multi-target scenarios. The system deals with a mass raid scenario and is designed to process situations with a varying number of targets and launch sites. The tracker receives input from the Environment Generator and Synthesizer module in terms of sensor scans and target information. The multiple target tracking system has two geostationary sensors which scan specific launch sites for missiles or targets launched from the surface of earth. The launch sites are specified in terms of latitudes and longitudes. The data from these two geostationary sensors are fed to two focal plane tracking (FPT) modules (2 dimensional tracking) at 5 second intervals. The focal plane tracking modules process this data using kinematic filtering algorithms and track pruning and prediction algorithms. The output of this module is an initial prediction of trajectories of launched missiles. This data is then fed to a three dimensional tracking system which uses the data from the two focal plane tracking modules to prune duplicate tracks (if any), extend existing tracks, prune bad tracks and initiate new tracks. The output of the system is a list of target trajectories.

2.1.1 Concurrent Multi Target Tracking (CMTT)

The Multi target tracker was initially developed at the California Institute of Technology under the Caltech Concurrent Computation Project [1]. It was implemented using the CUBIX programming model for embedded architecture (hypercube) viz. Mark III and CrOS III primitives. The CUBIX model is a hostless programming model where there is only one

program called a 'node' program which executes on every processor in the hypercube.

We modified the implementation of the tracker so it can be easily ported using existing parallel/distributed computing tools (EXPRESS, PVM, p4) on different platforms. To achieve this objective we developed a uniform structure of the multi target tracker [2]. We also developed an efficient implementation of the CMTT algorithm. In what follows, we discuss two parallel implementations of the CMTT system. In the first one, the sensors data are processed sequentially (CMTT-SSDP) while in the second one the sensors data are processed in parallel (CMTT-PSDP). Each scan of MTT begins with an existing track file and a new set of sensor reports. Existing tracks are extended using sensor reports which satisfy the gating criterion [1] of the track-split processor.

The concurrency in multi target tracking is achieved by using data parallelism. The data of the global track file, which has the details of processed data obtained from two geostationary sensors, is partitioned among the nodes involved in the CMTT. Each node executes the same code, but uses different data segments of the global track file. Every node has access to the full sensor reports file at every scan and it performs the sequential multi target tracking algorithm on its subset of the global track file.

The most time consuming step in CMTT-SSDP is the redistribution of global track file (step 2.1.3) and it is critical to achieve efficient concurrent implementation. Redistribution must be done such that all tracks ending at a given datum must be assigned to the same node in the next scan. This will reduce the number of duplicate tracks. Because of the irregular transfer of tracks between nodes during redistribution, the transfer of tracks among nodes is done using the *Crystal.Router* communication algorithm. It is an algorithm to redistribute the track file among all nodes involved in the parallel computation in $\log_2 N$ steps (where N is the number of processors).

Concurrent MTT with Parallel Sensor Data Processing (CMTT-PSDP) Figure 2 highlights the main tasks performed by the Concurrent MTT with the sequential sensor data processing (CMTT-SSPD) algorithm. Figure 2 also shows the modified version of this algorithm. In the CMTT-SSDP algorithm, the Do loop (for sensor 1 and sensor 2) in step 2.a is performed sequentially; i.e. first we do 2D tracking for sensor 1 and then perform 2D tracking for sensor 2. In this implementation, redistribution of the track file (step 2.1.3 in Algorithm CMTT-SSDP) is done between all of the nodes in the cube, for both sensor 1 and sensor 2. The performance of the CMTT can be improved by overlapping communication and execution. In this case, the 2D tracking of the two sensors' data is performed concurrently. As a result of processing the sensor data in parallel, the redistribution is done only between half of the nodes working on the same sensor data. This reduces the redistribution time considerably.

However, the 3D tracking is done on all nodes/processors.

In the algorithm CMTT-PSDP, after concurrent 2D tracking of both sensors, they must communicate the results with each other before 3D tracking can be initiated. After 2D tracking, each node in the same subcube has completed track and report files for the sensor data assigned to this subcube. Hence, instead of one processor sending results to every node, the communication occurs only between corresponding nodes in both subcubes. This allows us to overlap the communication between nodes and thus reduces its overhead. This exchange of results constitutes the extra overhead due to our new approach. But this extra overhead is insignificant when compared to the performance gained from overlapping the communication during track file redistribution.

After communicating the results, we reinitialize the cube environment to form one cube. The 3D tracking proceeds as in the algorithm CMTT-SSDP. We did not attempt to improve the performance of the 3D tracking because its execution time can be ignored when compared to the 2D execution time of the CMTT algorithm.

Algorithm CMTT-SSDP

1. Initialization
 - /* Initialize the parameters of sensors */
2. For I= 1,NO_SCAN Do
 - (a) For J= 1,NO_SENSORS Do
 - 2.1 2D Focal Plane tracking
 - 2.1.1 Compute focal plane data for sensor J.
 - 2.1.2 Extend existing tracks
 - 2.1.3 Track Redistribution
 - 2.1.4 Compute Focal Plane report
 - 2.1.5 Initiate new tracks
 - (b) 3D tracking (Combine results from both sensors)
3. print results

Algorithm CMTT-PSDP

1. Initialization
2. For I= 1,NO_SCAN Do
 - 2.1 Partition processors into NO_SENSORS subcubes
 - 2.2. 2D Tracking
 - If processor ID mod NO_SENSORS = J perform 2D tracking for sensor data J
 - /*similar to Algorithm CMTT-SSDP step 2.1 */
 - 2.3. Exchange the 2D results between processors working on different sensors
 - 2.4. Initialize the cube of N processors
 - 2.5. Perform 3D tracking as in Algorithm CMTT-SSDP step 2.2
3. print results

Figure 2: Algorithms CMTT-SSDP and CMTT-PSDP

2.1.2 Performance Results

In this section, we benchmark the implementation of the CMTT using different parallel/distributed tools. The main objective of this experimentation is to understand the issues related to porting compute intensive applications (with more than 32,000 lines of code) on parallel and distributed systems. Furthermore, we do need to determine the ideal problem size and type of platform (parallel or distributed computing environment). We benchmarked the CMTT system on two classes of computing environments: Distributed Computing Environment(SUN, IBM RS6000, IBM-SP1¹) and Parallel Computing Environment(CM5, iPSC 860).

Benchmarking CMTT on Cluster of Workstations On a distributed computing environment, the performance of the CMTT has been improved by increasing the number of processors up to a certain threshold. After that the performance starts deteriorating. Table 1 shows the comparisons of times taken on ATM(LAN), NYNET and Ethernet respectively for both CMTT-PSDP and CMTT-SSDP. We see that the performance of CMTT-SSDP on an ATM cluster is better than that on an Ethernet cluster. We don't see any improvement in CMTT-PSDP (for 2-nodes) because there is only nominal communication involved in the two node implementation. From Table 1 and Figure 3 we see that execution time deteriorates after two nodes and four nodes for CMTT-SSDP and CMTT-PSDP algorithms, respectively. It is clear from these figures that CMTT-PSDP performs much better than CMTT-SSDP because of the reduction in the the communication time associated with the redistribution of the track file.

In terms of platforms, IBM-SP1 out performed other distributed computing environments (SUN SPARC, IBM RS6000 and heterogeneous environment of SUN SPARC and IBM RS6000). For example, CMTT-PSDP implemented using PVM took 23.16 seconds on IBM-SP1 with four processors, whereas it took 65.56 seconds on four SUN SPARC workstations, 60.10 seconds on four IBM RS6000 workstations and 63.51 seconds on a heterogeneous environment of two SUN SPARC and two IBM RS6000 workstations. Furthermore, the PVM implementations outperformed other tools. However, for a small number of processors (say 2), the difference between tools is insignificant, while it is large for four or more processors.

Benchmarking CMTT on Parallel Computers When we implemented the CMTT system on parallel computers, we obtained consistent results with those of a distributed computing environment; the CMTT-PSDP version outperforms the CMTT-SSDP version. Also, the execution time reduces up to four nodes in the CMTT-SSDP version and up to eight nodes in

¹Configuration of IBM-SP1 uses dedicated Ethernet for interprocessor communication.

Table 1: CMTT performance on SUN IPCs

# of Nodes	CMTT-PSDP			CMTT-SSDP		
	ATM(LAN)	NYNET	Ethernet	ATM(LAN)	NYNET	Ethernet
1	180.75	180.75	180.75	179.56	179.76	179.56
2	107.50	107.56	108.39	140.67	143.84	161.81
4		75.34	91.38		162.67	190.81

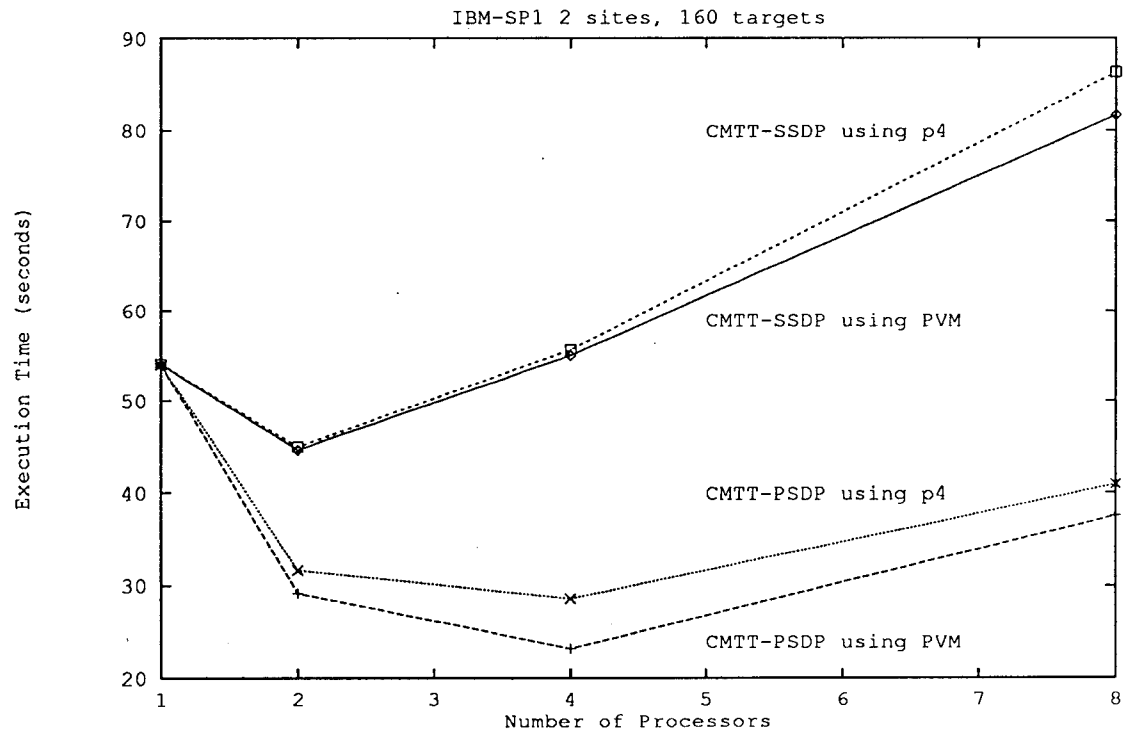


Figure 3: Performance Result of CMTT on IBM-SP1 implemented with p4 and PVM

the CMTT-PSDP version. Thus, the parallel computing environment works fine for a larger number of processors because the communication latency is less than that of Ethernet. For example, the CMTT-PSDP version using EXPRESS took 34.41 seconds on eight processors of iPSC 860, whereas the CMTT-PSDP version using PVM took 37.57 seconds on eight processors of IBM-SP1. When we compare the performance of the tracker on iPSC 860 and CM5, we found that the iPSC 860 implementation using EXPRESS performs better than CM5 using PVM.

2.2 Financial Modeling Application

2.2.1 Introduction and Problem Description

Financial modeling represents a promising industry application of high performance computing. In previous work, parallel stock option pricing models were developed for the Connection Machine-5 and DECmpp-12000 [5] [7], and later were ported on an IBM SP1 and a DEC Alpha cluster. These parallel models run approximately two orders of magnitude faster than sequential models on high-speed workstations. To further develop this application, a portable, workstation based, interactive visualization environment was developed for a heterogeneous computing environment. Application Visualization System (AVS) was used to integrate massively parallel processing, workstation based visualization, an interactive system control, and distributed I/O modules.

Using a stock option price modeling application as a case study, we demonstrated a simple, effective and modular approach to coupling network-based concurrent modules into an interactive remote visualization environment. Two prototype simulation on-demand systems are developed in which parallel option pricing models are locally implemented on two system configurations (two meta machines): one with two MPP machines, a 32-node CM5 and a 8K-node DECmpp-12000 [6]. The other has two distributed systems: an Ethernet-based IBM SP1 and a FDDI based network connecting a cluster of workstations [8]. These are coupled with an interactive graphical user interface over the NYNET ATM-based wide area network.

Stock option pricing models are used to calculate a price for an option contract based on a set of market variables (e.g. exercise price, risk-free rate, time to maturity) and a set of model parameters. Model price estimates are highly sensitive to parameter values for volatility of stock price, variance of the volatility, and correlation between volatility and stock price. These model parameters are not directly observable and must be estimated from market data. Using optimization techniques for model parameter estimation holds great promise for improving model accuracy.

We use a set of four option pricing models in this study. Simple models treat stock price volatility as a constant and price only European options (option exercised only at maturity of contract). More sophisticated models incorporate stochastic volatility processes and price American contracts (option exercised at any time in life of contract) [3] [4]. These models are computationally intensive and have significant communication requirements. The four pricing models are: BS – the Black-Scholes constant volatility, European model; AMC – the American binomial, constant volatility model; EUS – the European binomial, stochastic volatility model; and AMS – the American binomial, stochastic volatility model. Detailed descriptions about these four models can be found in [3] [4] [5].

Analytic models are useful tools in the financial market, but require expert interpretation. To further evaluate and optimize pricing models to run in a parallel computing environment, we combine high performance computing modules for real-time pricing with real-time visualization of model results and market conditions and a graphical user interface allowing expert interaction with pricing models. We envision a market expert using such a system to start and stop a set of models, adjust model parameters, and call optimization routines according to dynamically changing market conditions.

2.2.2 System Configuration and Integration

Two prototype systems for this application were developed and implemented on the NYNET. One focused on a meta computer consisting of two MPP machines and the other on distributed workstation clusters.

Configuration 1 — NYNET + CM-5 + DECmpp-12000 + Workstations Figure 4 is the system configuration of the first prototype interactive simulation-on-demand system for the option price modeling application, using the AVS/PVM framework proposed in [8] and utilizing the network infrastructure and distributed computing facility at NPAC.

The AVS kernel runs on a SUN10 workstation which acts both as an AVS server to coordinate data-flow and top-level concurrent control among remote modules, and as a network gateway which links the NPAC in-house host machines locally networked by an Ethernet to the regional end-user through the NYNET. The ATM-based link is built around two Fore switches that operate at 155 Mbps (OC3c) while the wide area network portion of the network operates at OC48(2400 Mbps) speed.

Our heterogeneous computing system for stock option pricing consists of four compute nodes, a home machine, and two file server machines. All workstations, including the front-ends of the DECmpp-12000 and CM-5, are connected by a 10MBit/second Ethernet based

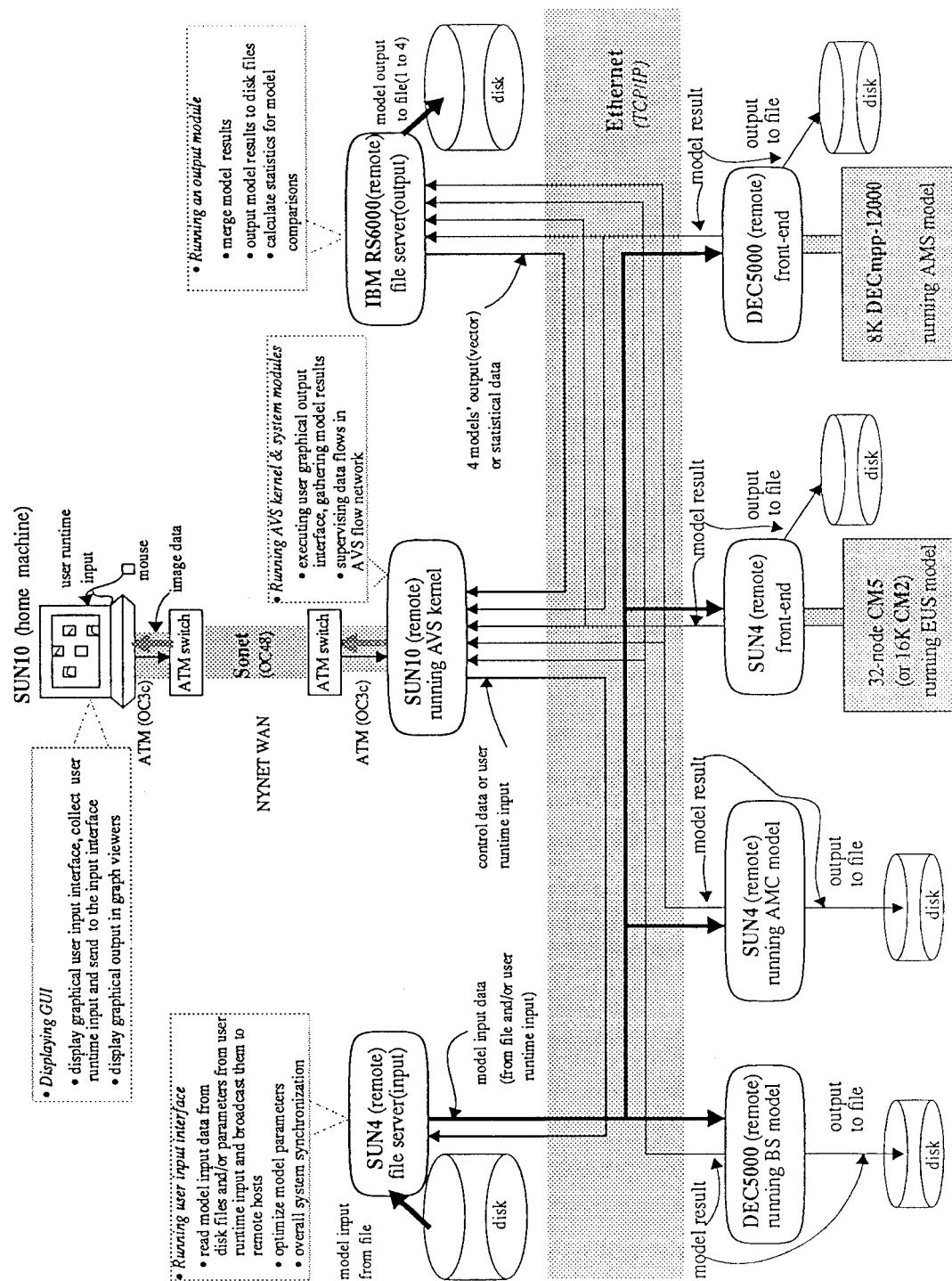


Figure 4: System Configuration 1 for the Financial Modeling on CM-5 and DECmp-12000

LAN.

The four option pricing models run on remote compute nodes: BS model on a DEC5000, AMC model on a SUN4, EUS model on a CM-5 and AMS on a DECmpp-12000(SX). Each remote compute node has its own I/O capability. Our DECmpp-12000 is a massively parallel SIMD system with 8192 processors. Each RISC-like processor has a control processor, forty 32-bit registers and 16 KBytes of RAM. All the processor elements are arranged in a rectangular two-dimensional grid and are tightly coupled with a DEC5000 front-end workstation. The theoretical peak performance is 650 Mflops. Our CM-5 is a parallel MIMD machine with 32 processing nodes. Each processing node consists of a SPARC processor for control, four proprietary vector units for numerical computation and 32 MBytes of RAM. The control node of the CM-5 is a SUN4 workstation. The theoretical peak performance is 4 Gflops. Sequential compute nodes include a DEC5000 and a SUN4. The DEC5000 performs at 6.8 Mflops and has 16 Mbytes memory. The SUN4 runs at 4.3 Mflops and has 32 Mbytes memory.

The user interface runs on a remote SUN4. This machine combines user runtime input (model parameters, network configuration) with historical market databases stored on disk and broadcasts this data to remote compute nodes. System synchronization occurs with each broadcast.

An IBM RS/6000 is used as a file server for non-graphical output of model data. In this application, model prices calculated at remote compute nodes and corresponding market data are written to databases for later analysis.

In summary, the heterogeneous computing system illustrated in Figure 4 provides distributed computing, distributed memory, and distributed input/output for the stock option pricing application.

Our heterogeneous computing system integrates diverse functions – computation, visualization, and system control over a diverse set of hardware. We use a mix of programming languages on the remote compute nodes – Fortran77 on the DEC5000, C on the SUN4, CM-Fortran on the CM-5, and MPL (data parallel C) on the DECmpp-12000. AVS integrates visualization, networking functionality, and computation. At the operating system level, all remote modules are compiled and linked as stand-alone programs. Input and output ports are defined in modules by the programmer using specific library routines provided by AVS. Each module represents a process. Inputs and outputs between remote modules are implemented via socket connections.

There are two sources of input data: historical market data read from disk files and runtime input of model parameters by the user through a GUI. Output from all four models is rendered in a graphics window, displayed numerically in a shell window, and written to a database by the file server.

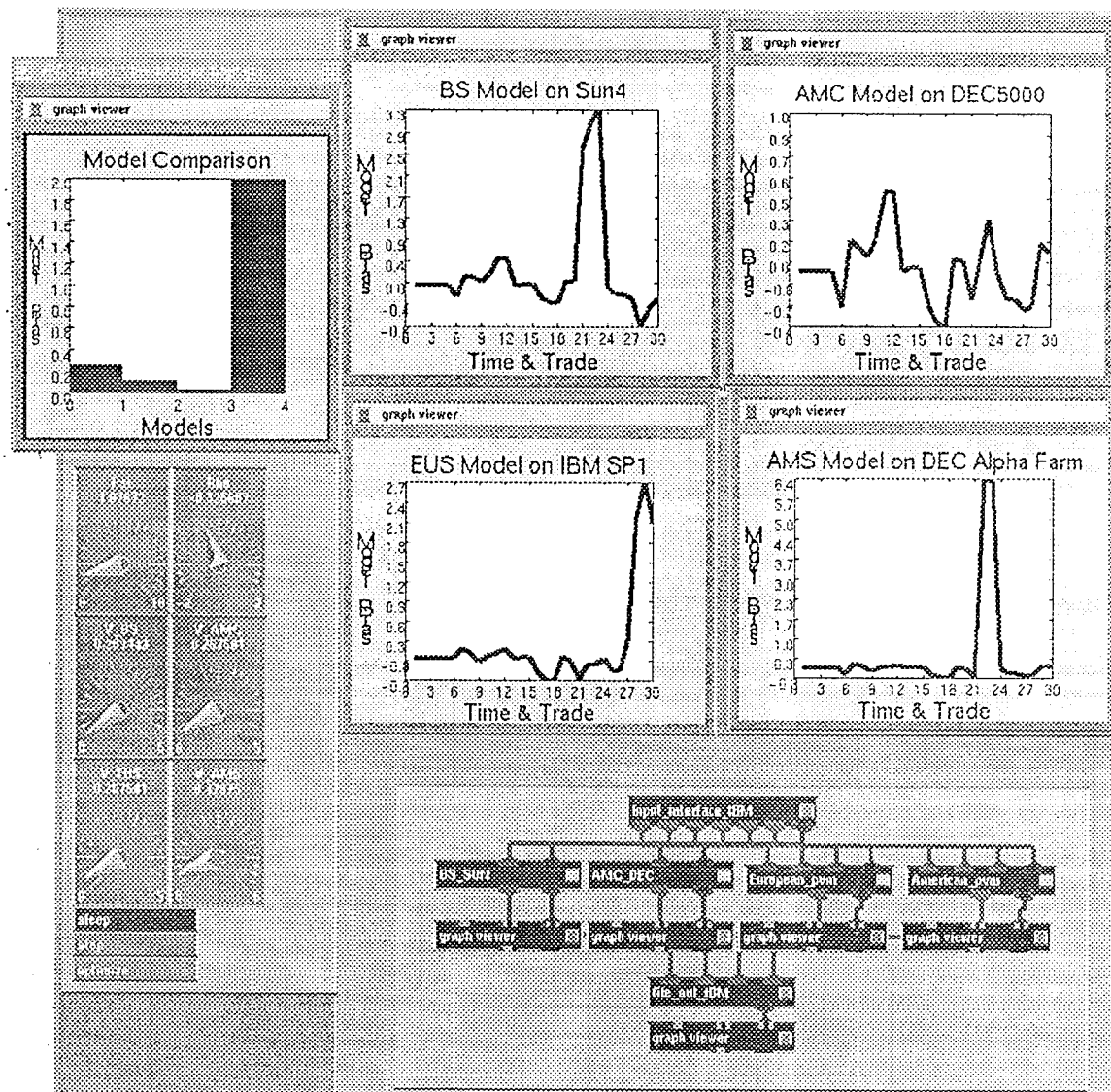


Figure 5: The Graphical User Interface on the Home Machine

Figure 5 illustrates the GUI for managing user runtime input and output and the system configuration. Runtime input includes user defined model parameters and system execution styles. Outputs include 2-dimensional displays of model and market prices calculated by the compute nodes. The system configuration includes choice of pricing models, network configurations and interface layouts.

Pricing models are extremely sensitive to model parameters for implied volatility, variance of stock volatility and correlation between stock price and its volatility. These parameters may be read from data files (historical estimates), calculated just prior to running the pricing

model (by optimization), or defined at run time (expert user).

Configuration 2 — NYNET + IBM SP1 + DEC Alpha Farm + Workstations

Figure 6 is the system configuration of the second prototype interactive simulation-on-demand system for the option price modeling application, using the AVS/PVM framework proposed in [8] and utilizing the network infrastructure and distributed computing facility at NPAC.

The AVS kernel runs on a SUN10 workstation. This workstation acts both as an AVS server to coordinate data-flow and top-level concurrent control among remote modules and as a network gateway. The gateway links the NPAC in-house host machines, locally networked by an Ethernet, to the regional end-user through the NYNET.

The two parallel pricing models (EUS model and AMS model) are implemented in PVM and run respectively on an 8-node IBM SP1, networked by an Ethernet at the time of evaluation, and a 8-node DEC Alpha cluster inter-connected by an FDDI-based GIGAswitch. They are coupled under the proposed AVS environment with the other two sequential simple models (BS model and AMC model) running on a SUN4 and a DEC5000 workstation, respectively. The nodal processor of SP1 is the IBM RISC/6000 processor running at 62.5 MHz. It is one of the most powerful processors available. The DEC Alpha farm consists of 8 Alpha model 4000 workstations which are supported by a high performance networking backbone of dedicated, switched FDDI segments. The GIGAswitch provides full FDDI bandwidth and low latency switching to every workstation in the farm.

While displayed on the end-user's home machine, a user interface actually runs on a remote SUN4 which combines user runtime input (model parameters, network configuration) with historical market databases stored on disk and broadcasts this data to remote compute nodes. Top-level system synchronization occurs with each broadcast.

An IBM RS/6000 is used as a file server for non-graphical output of model data. In this application, model prices calculated at remote compute nodes and corresponding market data are written to databases for later analysis.

All model outputs are graphically displayed on the end-user's home machine (a SUN10) in AVS graph viewers. Figure 5 is the user interface showing the simulation control panel(left), model output windows(top) and the flow network(bottom).

2.2.3 Performance Analysis

The timings for one trade of the parallel option models on various platforms is given in Table 2. Note:

- The timing data is measured when the level of the binomial tree is 17.

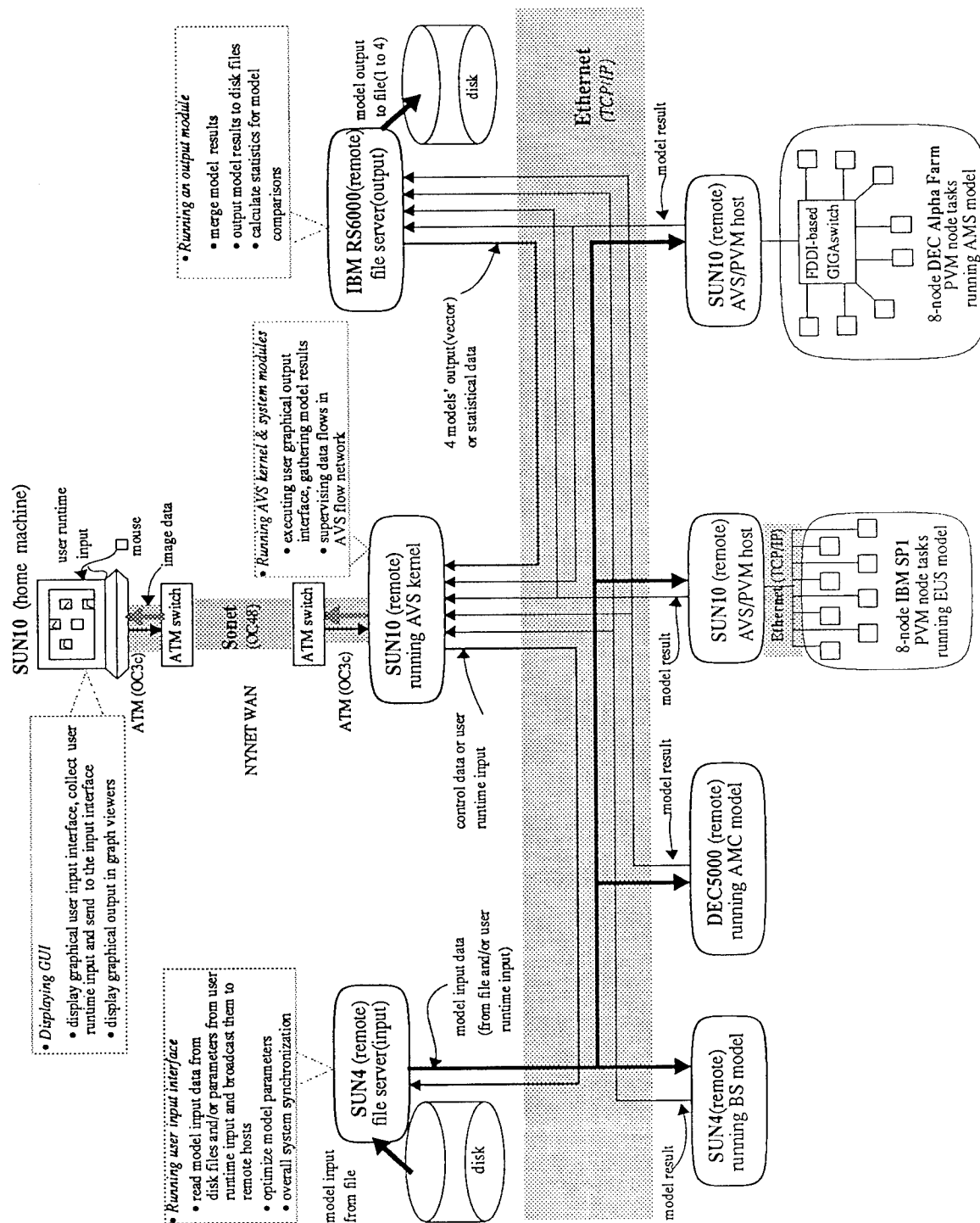


Figure 6: System Configuration 2 for the Financial Modeling on IBM SP1 and DEC Alpha Farm over NYNET

- On MIMD machines, the two models do not depend heavily on communication, but depend strongly on the node performance of the parallel systems. On SIMD machines, the models depend more heavily on communication. Different algorithms are used on MIMD (with an explicit message passing paradigm) and on SIMD (with Fortran90 data parallel paradigm) systems.
- EUS — EUropean Stochastic volatility binomial model;
- AMS — AMerican Stochastic volatility binomial model.

2.2.4 Conclusion

The financial modeling application implemented on the NPAC supercomputer facility and demonstrated over the NYNET gives a promising application of simulation-on-demand on the information superhighway. It combines the high-performance computing at a supercomputer center like NPAC with a high-bandwidth, wide area network like NYNET, for high-speed remote access and distributed computing.

We are exploring a new software framework in this area and plan to apply the integration technique described in this work to other InfoMall applications. We plan to add a network user interface, Mosaic, on top of the AVS framework. It is a distributed hypermedia software tool from NCSA to support InfoVision Simulation-On-Demand projects over the ATM-based wide area network. We believe that methodologies and tools for information integration will play an increasingly important role with the adoption of HPCC technologies in industry.

2.3 Electromagnetic Scattering

2.3.1 Introduction and Problem Description

Electromagnetic scattering (EMS) simulation is an important computationally intensive application within the field of electromagnetics. Advances in high performance computing and communication (HPCC) and the data visualization environment (DVE) provide new opportunities to visualize real-time simulation problems such as EMS which require significant computational resources.

Scientific visualization has traditionally been carried out interactively on workstations or in post-processing or batch on supercomputers. With advances in high performance computing systems and networking technologies, interactive visualization in a distributed environment becomes feasible. In a remote visualization environment, data, I/O, computation and user

Table 2: Timing for One Trade of the Parallel Option Pricing Models on Various Platforms

Platform	Machine size	EUS (sec.)	AMS (sec.)	Speedup	
				EUS	AMS
SUN10(seq.)	1	1.087	1.186		
SUN4(seq.)	1	2.07	2.31		
SUN IPC(seq.)	1	4.05	4.25		
CM-5(with VU)	32		0.025		
DECmpp-12000	8192	0.075	0.045		
CM-2	8192		0.05		
Alpha+Gigaswitch (PVM3)	1	0.469	0.553	1	1
	2	0.239	0.279	1.96	1.98
	4	0.130	0.151	3.61	3.67
	8	0.089	0.099	5.27	5.59
IBM-SP1+Ethernet (PVM3, EUI/IP)	1	0.505	0.568	1	1
	2	0.260	0.290	1.94	1.96
	4	0.145	0.160	3.48	3.55
	8	0.094	0.110	5.37	5.16
IBM-SP1+HPswitch (PVM3, EUI)	8	0.0602	0.0663		

interaction are physically distributed through high-speed networking to achieve high performance and optimal use of various resources required by the application task. Seamless integration of high performance computing systems with graphics workstations and traditional scientific visualization is not only feasible, but will be a common practice with real-time application systems.

In this work, an integrated interactive visualization environment was created for an EMS simulation. It coupled a graphical user interface (GUI) for runtime simulation parameters input and 3D rendering output on a graphical workstation with computational modules running on a parallel supercomputer and two workstations. The Application Visualization System (AVS) was used as integrating software to facilitate both networking and scientific data visualization. This interactive visualization environment can be run from remote and distributed users via the NYNET wide area network with sufficient bandwidth to support run-time simulation and model parameters steering.

Electromagnetic scattering (EMS) is a widely encountered problem in electromagnetics with important applications in industry such as microwave equipment, radar, antenna, aviation, and electromagnetic compatibility design. Figure 7 illustrates the EMS problem we are modeling. Above an infinite conductor plane, there is an incident EM field in free space. Two slots of equal width on the conducting plane are interconnected to a microwave network behind the plane. The microwave network represents the load of waveguides; for example, a microwave receiver. The incident EM field penetrates the two slots which are filled with insulation materials such as air or oil. Connected by a microwave network, the EM fields in the two slots interact with each other, creating two equivalent magnetic current sources in the two slots. A new scattered EM field is then formed above the slots. We simulate this physical phenomena and calculate the strength of the scattered EM field under various physical circumstances. The presence of the two slots and the microwave load in this application requires simulation models with high performance computation and communication. Visualization is very important in helping scientists to understand this problem under various physical conditions.

In previous work, data parallel and message passing algorithms for this application were developed to run efficiently on massively parallel SIMD machines such as Connection Machine CM-2 and DECmpp-12000, and MIMD machines such as the Connection Machine CM-5 and iPSC/860. The data parallel algorithms run approximately about 400 times faster than sequential versions on a high-speed workstation [9]. Parallel models on high performance systems provide a unique opportunity to interactively visualize the EMS simulation in real-time. This problem requires a response time of the simulation cycle that is not possible on conventional hardware.

Figure 7 also shows physical parameters of the electromagnetic scattering problem.

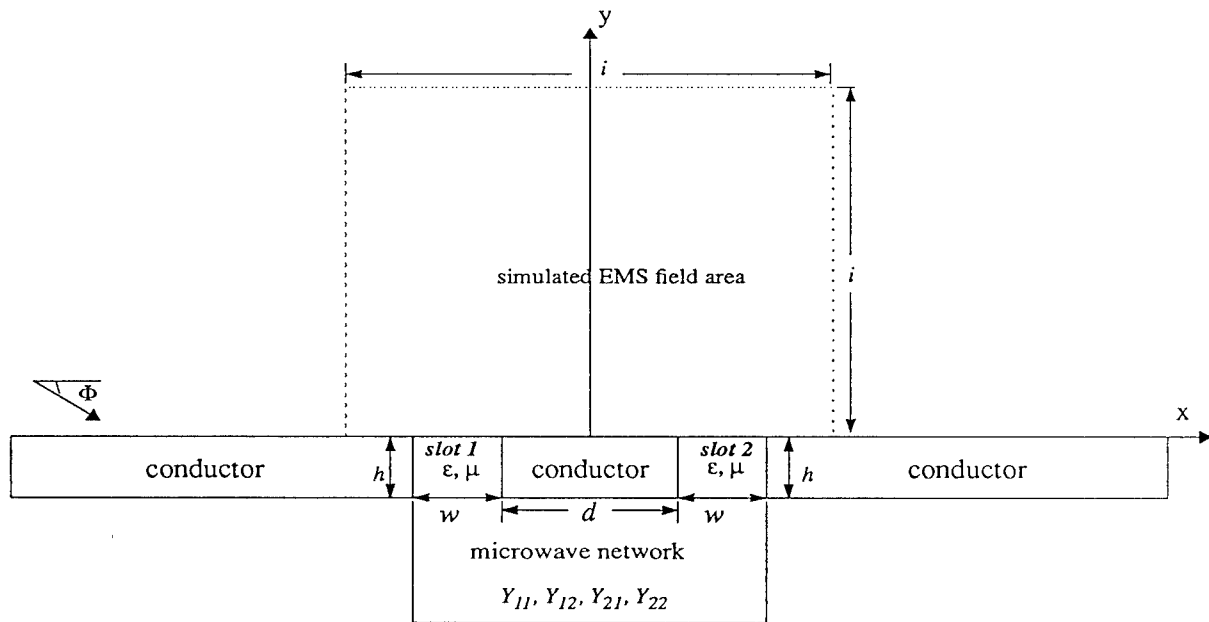
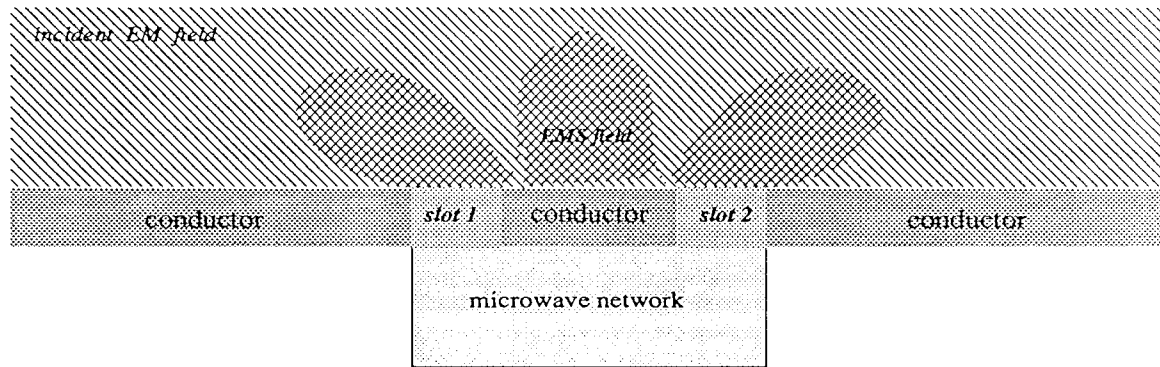


Figure 7: Profile of the electromagnetic scattering problem

2.3.2 System Configuration and Integration

Figure 8 illustrates the system configuration and module components distributed over the network connecting three high-end workstations and a supercomputer Connection Machine 5. The network is a 10 MBit/s Ethernet-based local network. Commercially available AVS software is used to provide sophisticated 3D data visualization and system control functionality required by the simulation. We use AVS to facilitate high level networking and data transfer among visualization and computational modules on different machines in the system.

AVS provides a data-channel abstraction that transparently handles type-conversion and module connectivities. This software system is optimized for data movement by using techniques such as shared memory message passing among modules on the same machine. Message passing occurs at a high level of data abstraction in AVS. This approach helps to make optimal use of both the high performance computing resources and the rendering capabilities of the local graphical workstation. The transparent networking capabilities of AVS open up possibilities for visualization far beyond traditional graphics capabilities.

The local machine in our system is an IBM RS/6000 with a 24-bit color GTO Graphics Adaptor. An AVS coroutine module (in C) on the local machine serves as a graphical input and system control interface to monitor and collect user runtime interaction with the simulation through keyboard, mouse and other I/O devices. The AVS kernel also runs on the local machine, coordinating data flows and control flows among AVS (remote) modules in the network.

The computationally intensive modules of this application are distributed to a CM5, a MIMD supercomputer, which is configured as 32 processing nodes at NPAC. Each processing node(PN) of the CM5 consists of a SPARC processor for control and non-vector computation, four vector units for numerical computation and 32 MB of RAM. It also includes a Network Interface chip which gives the node access to the CM5 internal Data Network and Control Network. The two internal networks connect all the PNs with a control processor(CP) which runs a custom version of SunOS on a SPARC host. Two Sun SPARC workstations are used in our distributed visualization environment to run the computational modules with modest communication requirements.

All modules other than those on the local machine are implemented as AVS remote modules. Their input/output ports are defined by specific AVS libraries for receiving/sending data from/to other (remote) modules via socket connections. This configuration allows the interrupt driven user interface input mechanisms and rendering operations to be relegated to the graphical workstation, while the computationally intensive components run on the CM5 coupled with the two workstations. This distributed simulation environment implemented in

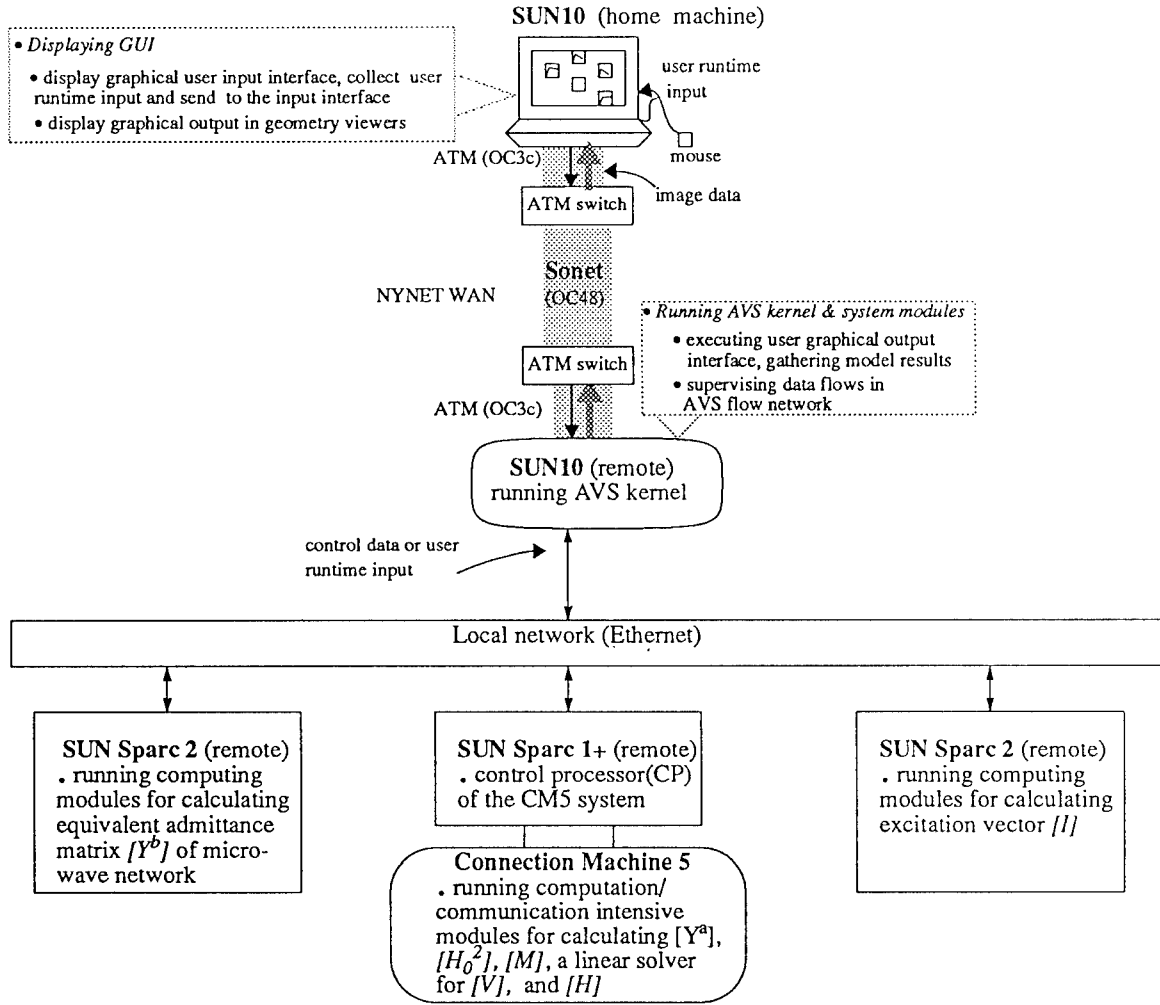


Table 1: Timings of calculations and communications (in second)

Module name	Calculation time	Communication time with 'Input-interface-IBM'	Communication time with 'EM-all-CM5'	Communication time with 'EM-3D-IBM'
EM-1-SUN	0.1 (Sun)	0.02	0.045	
EM-2-SUN	0.6 (Sun)	0.02	0.5	
EM-3-CM5	1.8 (CM5) 1260 (Sun)	0.02	0.0	
EM-all-CM5	2.1 (CM5) 120 (Sun)			0.5
AVS Rendering Modules	3.5 (550x580 window)			

Figure 8: System Configuration for Electromagnetic Scattering

AVS provides a transparent mechanism for using distributed computing resources along with a sophisticated user interface component that permits a variety of interactive, application-specified inputs.

The graphical user interface shown in Figure 8 includes a main control panel, three individual input panels, and a 3D rendering window.

The main control panel provides the user parameters input and simulation control at run-time. There are seven dial widgets representing simulation parameters used by all computing modules on the three remote machines, and a control button for starting a new simulation cycle(see lower left in Figure 8).

2.3.3 Performance Analysis

Our experiments show that under a typical working environment(only 0.5 MBits/s of the Ethernet's 10 MBits/s capacity are available), a complete simulation cycle takes about 8 seconds. This response time is quite satisfactory for this application. Table 1 in Figure 8 lists timing data of major system components. For comparison, timings of a sequential implementation on a SUN4 workstation of the two parallel modules are also given in the Table.

2.3.4 Conclusion

The performance limiting factors in this system are the sequential rendering operations on the local machine, and high-latency data transfer over the local area network due to multiple communication protocol layers. We focus here on the feasibility of applying a high-level distributed programming environment to a real application problem which requires both sophisticated 3D data visualization and high performance computing.

2.4 Parallel JPEG

2.4.1 Problem Description

Advances over the past decade in many aspects of digital technology - devices for image acquisition, data storage, and bitmapped printing and display - have brought about many applications of digital imaging. However, these applications tend to be specialized due to their relatively high cost. The main problem with digital imaging applications is that a vast amount of data is required to represent a digital image directly. This problem magnifies when we have to transfer images in real time such as in multimedia applications like Video-on-Demand. For example, if an application requires 25 frames/second and each frame is 640x480 pixels with 24 bits per pixel for color information, then it needs a network with a bandwidth of 184

Mbits/second, which is not provided even by high-speed networks like ATM and FDDI. Thus, because of high storage and transmission costs, the use of digital images has not been widely used. This problem is solved by image compression technology where original uncompressed images are compressed to 1/10-1/50 of their original size without affecting image quality.

JPEG (Joint Photographic Experts Group) is emerging as a standard for image compression. This is a standard image compression method which enables interoperability of equipments from different manufacturers. The JPEG standard aims to be generic to support a wide variety of applications for continuous-tone images. The JPEG standard includes two basic compression methods, each with various modes of operation. A DCT (Discrete Cosine Transform) based method is specified for *lossy* compression and a predictive method for *lossless* compression. JPEG features a simple lossy technique known as the *Baseline* method. It is a subset of the other DCT-based modes of operation.

In multi-media applications like video-on-demand, the speed at which compression and decompression are performed is very critical. Hence, sequential compression algorithms may not be suitable for such real-time applications. Efficient parallel compression/decompression algorithms are needed for these types of applications. We have implemented a parallel JPEG image compression/decompression method in a distributed computing environment.

2.4.2 System Environment and Configuration

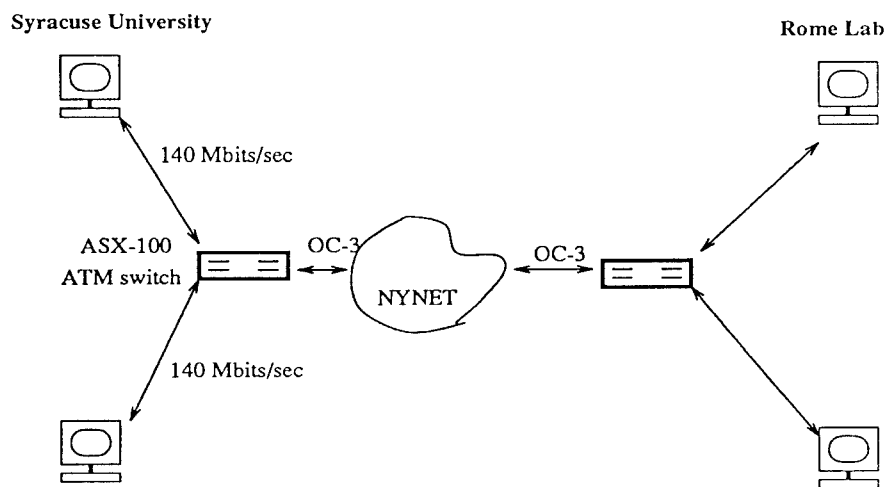


Figure 9: System Environment for JPEG

We have compared the performances of this application on different platforms which included a cluster of workstations connected by a Local ATM network, a Wide area ATM network (NYNET) and Ethernet. The LAN ATM network consists of two SUN IPXs directly connected

to a Fore ASX-100 local ATM switch. Both SUNs are equipped with a Fore SBA-200 ATM network interface on the SBus. Fore's SBA-200 uses an Intel i960 as an onboard processor. The i960 takes most of the AAL and cell related tasks including the SAR (Segmentation and Reassembly) functions for AAL 3/4 and AAL 5 and cell multiplexing. The physical media is the 140 Mbits/sec TAXI interface (FDDI fiber plant and signal encoding scheme). A part of NYNET which we have used consists of 2 SUN IPXs at Syracuse University and 2 SUN SPARCstations at Rome Labs, connected by the NYNET testbed as shown in Figure 9. The Ethernet set-up consists of SUN ELCs connected by a 10Mbits/sec Ethernet network.

2.4.3 Implementation Description

This implementation of JPEG compression/decompression uses a DCT-based *lossy* compression method. The user can trade off output image quality against compressed file size by adjusting a compression parameter.

Since the JPEG sequential algorithm performs the image compression line by line, where compression of each line is independent of any other line, we could take advantage of the inherent data parallelism. So, we have used the data parallel model while implementing JPEG on a cluster of workstations. The image to be compressed or decompressed is divided into N equal parts (where N is the number of processors) by the master process and are shipped to the remaining processors. Then, each processor performs the sequential JPEG compression algorithm on its portion of the image. After compression, the processors send the compressed image to another set of N processors which perform the decompression. Once decompression is done, the results are sent back to the master process which combines them into one image. So, basically this algorithm involves five stages: distribution of uncompressed image by the master process, compression of the image by a set of N processors, shipping of compressed image to another set of N processors, decompression of the image by these processors, and displaying the image by the master process after receiving all the parts of the decompressed image.

2.4.4 Performance Results

We demonstrated the performance of a distributed application when a high speed ATM network is used as the interconnection network. We compared the performance of this application when it is run over an ATM network with the performance when it is run over Ethernet.

The results of the performance for an image of size 596KB over NYNET are shown in Table 3. The times shown indicate the total time (in seconds) taken by all five stages of the JPEG compression/decompression algorithm. The size of the image after compression was

Table 3: JPEG performance

# of Nodes	Total Time (sec.)		
	ATM(LAN) ²	NYNET	Ethernet ³
1	5.05	5.05	8.26
2	5.10	3.81	9.06
4		2.27	5.59

32KB; a reduction of more than 18 fold. This reduces both of the problems of a digital image application: transmission and storage cost. We didn't see any performance improvement with two nodes because of the way algorithm is implemented. Only half of the processors are active at any time. Hence time taken by two nodes is slightly worse than one node because of interprocessor communication. The performance improvement over NYNET for two processors is due to the fact that the machines at Rome Labs are faster than the ones at Syracuse University. We will see the performance improvement with two nodes if all the nodes are active at the same time. Since this application was originally written for demonstration purposes, it requires some modifications to improve its' performance before it can be used for practical multimedia applications.

2.5 Syracuse Language Systems

2.5.1 Problem Description

Most of the multimedia software that runs on PCs is being distributed over a Compact Disk medium. Consequently, to access this multimedia software, all PCs must have CD-ROM drives and each user requires one copy of the CD-ROM software. In this project, we investigated the development of a multimedia server that can store all the CD-ROM software and have PCs access this server over a high speed network such as the NYNET. A proof of concept has been demonstrated by porting the *TriplePlay* multimedia software developed by Syracuse Language Systems (SLS) for teaching languages, to a server accessible from remote multiple PCs. We have used PC-NFS, which is a PC version of the SUN Network File System (NFS), to transparently access the files of the UNIX file system. Further, we have demonstrated that our approach is general and can be applied to any other multimedia software distributed on CDs.

²Machines used are IPXs

³Machines used are ELCs

2.5.2 System Environment and Integration

PC-NFS: Software developed by Sun Microsystems enables personal computers running MS-DOS to share information and resources with workstations, minicomputers and mainframes that run different operating systems including UNIX and VMS. This sharing is provided transparently in a similar manner to the sharing of files among a cluster of workstations running NFS.

By using PC-NFS, the remote file systems are mounted on local disk drives and remote printers are mounted on three parallel printing devices that DOS recognizes; LPT1, LPT2, and LPT3. Once the remote file systems or printers are mounted they can be accessed as though they are separate local drives or local printers running under the DOS environment.

SLS Software Syracuse Language Systems is a company which develops multimedia software for language education. In this project, we have used their "Playing with Language" series to demonstrate Education on Demand. Their *TriplePlay* software helps the users learn a foreign language. *TriplePlay* uses enhanced graphics to display objects of different complexity and sizes. When a single object is selected, the software pronounces the word corresponding to this object. Further, *TriplePlay*'s conversational features help users to learn, understand and speak parts of realistic dialogues and conversations.

Porting SLS and Current Configuration When many users want to share a CD-ROM based multimedia software, they need to have CD-ROM drives and each one should have a copy of the CD-ROM software. Porting the software to a server accessible over a high speed network reduces the cost as well as access time of the multimedia software. We have copied all the files of the software from the CDs to a disk on the server. Then, we installed PC-NFS on all the PCs that need to access the software and mounted the directory containing the files to a local drive. This method of sharing a multimedia software by multiple PCs is not restricted to SLS software but can be used with any other CD-ROM software.

Figure 10 shows the current configuration of the SLS project. In this configuration, the server exports some part of its file system to the PCs so that they can access the server transparently using PC-NFS. From the PC side, they have to install a communication driver and PC-NFS software in order to access the server. After installing PC-NFS, the exported file system should be mounted by the PC-NFS. Consequently, the PC users can access the UNIX based server's file system in exactly the same way it accesses a local disk. *TriplePlay* which is running on a PC can access its files, stored at the server as if they are local, through the network (ethernet) and thus eliminate the need for CD-ROM software and drive. This

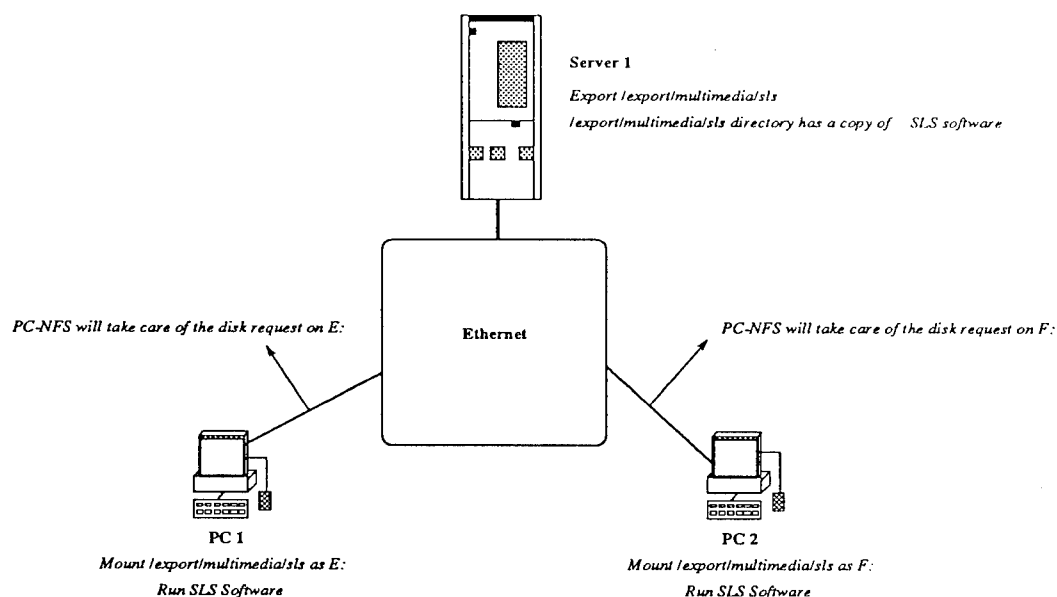


Figure 10: Current Configuration of SLS Project

represents an interesting approach to deliver information on demand to a large number of PC users.

2.5.3 Performance Issues

The use of a high speed network is critical to the development of a large scale multi-media server. In the SLS multimedia server, the Ethernet bandwidth could be a bottleneck when a large number of PCs access this server simultaneously. The high bandwidths of NYNET makes it an ideal network to implement this type of multimedia server. Moreover, the most critical aspect of this server is its storage capacity. For such a server to store hundreds of CD-ROM multi-media programs, it needs storage space on the order of 60 Gbytes (each CD-ROM capacity is about 600 Mbytes). To reduce the disk space, one can use data compression techniques. However, this approach must be studied carefully because it increases the access time of the multimedia server. The NYNET multimedia server can universally be accessed if it is connected to an ISDN network. We are currently investigating how the ISDN network can be used to access such a multimedia server.

2.6 Fractal Imaging

2.6.1 Problem Description

Fractal geometry is an interesting field of mathematics that can be used to represent irregular or fragmented features not representable with Euclidean geometry. A **fractal** is a rough or fragmented geometric shape that can be subdivided in parts, each of which is (approximately) a reduced-size copy of the whole. Fractals are generally self-similar and independent of scale. There are many mathematical structures that are fractals; examples include Sierpinski triangle, Koch snowflake, Peano curve, Mandelbrot set, and Lorenz attractor. Fractals also describe many real-world objects, such as clouds, mountains, turbulence, and coastlines, that do not correspond to simple geometric shapes.

One of the most popular sets of fractal images is the Mandelbrot Set. This Set is a fractal geometry that consists of a series of points in the complex plane which satisfy a certain condition. The Mandelbrot set is the set of all complex c such that iterating $z \rightarrow z^2 + c$ does not go to infinity (starting with $z=0$). These sets in general are not strictly self-similar. The tiny copies of the Mandelbrot set are all slightly different, mainly because of the thin threads connecting them to the main body of the Mandelbrot set. However, these are *quasi-self-similar*. These sets can be effectively partitioned and replicated using the data-parallel model. In addition, computation can efficiently be distributed using a client-server architecture because of the minimal data that needs to be passed between processes.

2.6.2 System Environment and Configuration

The experimental environment for Fractal imaging consists of SUN ELCs interconnected by Ethernet. We have used the Motif library for displaying fractal images. As Motif libraries were not installed on the machines connected by the ATM network, we could not run this application over the ATM LAN and NYNET.

2.6.3 Implementation Description

Because of the compute intensive nature involved in generating fractal geometries, the ability to display fractal images has always had a direct correlation to the computational power of available digital computers. In addition to fast machines, the availability of high speed networks connecting workstations has allowed fractal generators to use another performance enhancing technique: distribution of computational workload, in other words, distributed computing.

Table 4: Mandelbrot performance

# of Nodes	Total Time (seconds)
1	511.29
2	251.28
4	145.11
8	81.25

Because of inherent data parallelism in fractal generation, the model used for parallelization is the data parallel model (i.e. each node was used to calculate a portion of the image). This is done by dividing the image in N equal parts where N is the number of processes. Each process executes in parallel and is responsible for all the pixels in its part. Since the calculation of each pixel is independent of other pixels, there is very little inter-processor communication overhead. Due to this fact, we were able to achieve good speedups.

2.6.4 Performance Results

The performance results of this fractal image application over a cluster of SUN ELCs connected by Ethernet are shown in Table 4. We were able to get good speedups because of the compute intensive nature of the program. The communication overhead is very little when compared to the computations performed by this application. Compute intensive applications with very little communications are not likely to benefit much from high-speed networks such as NYNET. This is due to the fact that we are not using the high bandwidth provided by the network. We couldn't run this application on the workstation cluster connected ATM network, but we expect the speedups to be almost the same.

3 Evaluations of NYNET Enabling Technology

3.1 Mosaic Server

NCSA Mosaic is a distributed hypermedia system designed for information over the Internet. It provides a unified, intelligent graphical user interface to various protocols, data formats, and information archives used on the Internet and enables powerful methods for discovering, using, and sharing information. Mosaic is the public domain software developed by the National Center for Supercomputing Applications. It uses a client/server model for information distribution. Units of information (documents), sent from servers to clients, may contain plain text, formatted text, images, sound, video and hyperlinks to other documents anywhere on

the Internet. Mosaic supports interfaces to Gopher, FTP, WAIS, Techinfo, TeXinfo, finger, Whois and other Internet data resources. Mosaic clients can be installed and used on almost any modern Unix-based graphics workstation (SunSparc, IBM RS/6000, DEC 5000, Alpha, Silicon Graphics IRIS). The Macintosh and Microsoft Windows clients also exist. To allow interaction with a wide variety of data formats (JPEG, XWD, TIFF, RGB, MPEG, DVI, PostScript etc.), Mosaic relies on a number of external viewers: xv, showaudio, mpeg-play, xdvi or ghostview.

Mosaic is used as a user-friendly interface to the NPAC on-line information services which include:

- database of information pertinent to NPAC, which specializes in High Performance Computing and Communications, parallel processing, distributed computing, computational science, education, and technology transfer through the InfoMall program.
- distribution of demo software and NPAC software products in such areas as simulation and video on demand. The Mosaic software is extensible and supports on-line demo sessions started remotely on computers in NPAC. To do this we installed the Mosaic HyperText Transfer Protocol (HTTP) server and Mosaic clients on several environment platforms (Sun, SGI, DEC, Alpha, IBM RS, Microsoft Window, Apple MacIntosh).

The NPAC WWW Server contains the following information:

- announcements (what's new in Web, important events)
- general information about NPAC (NPAC organization, contact addresses, phone list, home pages of NPAC researchers, NPAC seminars, administrative documents and forms, local news server link, FTP server link, an overview of NPAC)
- description of research projects, divided into several categories: simulation and parallel algorithms, parallel languages and compilers, parallel programming tools and software, software integration, InfoVISion (Information, Video, Imagery, and Simulation on Demand) and education.
- computing facilities
- how to use NPAC's computing facilities
- technical reports and papers
- HPCC software and information

- the InfoMall technology transfer program
- education programs
- related HPCC projects, organizations, and information
- Syracuse University Web servers

The official NPAC Web server can be accessed under the following URL: <http://www.npac.syr.edu/>. We have developed an HTTP server to run NYNET demonstrations from a Mosaic front end user interface. These demonstration programs include:

- A Grand Challenge Tornado Prediction Model
- Chemistry Transport in the Atmosphere
- Electromagnetic Scattering Simulation
- Radar Cross Section Simulation
- Stock Option Pricing Model

The results of these simulations on demand are accessible over NYNET through sophisticated network-based user interface Mosaic software. This is an example of future “simulation on demand” products for home and school markets, with high-speed networks providing the essential link between High Performance Distributed Computing facilities and end users. The currently available demonstrations are incorporated into an AVS based graphical interface to provide the three dimensional rendering and interactive model control. The active role of the user is supported by the system. The simulations are fully interactive, so the user can change some parameters dynamically. For example, users can learn the meteorology of tornados by observing the simulation and by experimenting with pressure and temperature changes over the network. All of these simulations are very demanding in terms of required high-performance resources. Calculations are performed in real time on high-performance computers (SP2, CM5, DECmpp, Alpha cluster) at NPAC.

The simulation-on-demand programs are started after clicking on a hyperlink in a Mosaic interface. The front-end interface used to launch these demonstrations is based on the interactive fill-out forms and post-script execution under the control of a Mosaic server daemon. Support for fill-out forms inside a Hyper Text Markup Language (HTML) document enables usage of text entry areas, option buttons, radio buttons, option menus, scrolled lists and image maps.

The Mosaic client/server software is also used as an interface to the experimental CNN Newsource online videoclips. We have captured and digitized a number of short movies. Two different setups have been installed: Sun-based Parallax video cards and SGI Indy workstations. The cluster of workstations used for the video-on-demand demonstrations is linked via an ATM LAN supported by the FORE ASX-100 switch. ATM links are used to support compressed video data delivery to the browsers via the NFS protocol.

We have investigated:

- mapping file extensions to the MIME types
- mapping MIME types to external viewers
- execution of shell scripts and post-scripts via hyperlinks.

The basic functionality of the VOD demonstration is provided by two independent software modules: digital video browsers, implemented at NPAC, and a Mosaic-based user interface. The Mosaic interface to VOD has been chosen for compatibility with other InfoVision projects, such as simulation on demand and InfoSchool.

3.2 Communique Software

Communique! is the video conference tool developed by InSoft, Inc., that we have used for demonstrating how the ATM networks like NYNET provide the high bandwidths required for these applications. Communique! integrates the multimedia aspects of graphics, audio, video, text and native application files into a real-time, on-line conference.

Communique! contains a suite of easily maneuvered iconic tools to guide the user through defining and initiating an on-line, real-time conference with fellow workgroup members. Like any conference room, the Communique! Virtual Conference Room contains tools that help people exchange ideas and information. Audio Conferencing, Video Conferencing, a Shared White Board, and shared Text Tools are just some of the tools integrated in the application and available to the conference participants. Communique! can be used for real-time reviews of projects, simultaneous, concurrent engineering activities, on-line presentations, training, remote support, customer service applications, long distance interviews, and more. Communique! has several supporting tools to facilitate video conferencing activities. These tools include:

- The Audio Tool which allows users to talk with one another freely.

- The Shared White Board allows the users of Communique! to distribute a blank text screen that acts as a posting board for conference user's comments.
- The Shared Raster White Board allows users of Communique! to distribute a raster image to others in the conference and simultaneously make markups on this image.
- The Text Tool lets the users incorporate any textual data into the conference.
- The graphics tool allows users to share Sun Raster data with other conference members.
- The Video Tool of Communique! allows the users to work with any video input to capture still video images to be shared as graphics in the conference.
- The TV Tool enables a Communique! user to conduct a real-time Video Conference from their desktop.

3.2.1 Discussion

The Communique! software tool provided us with the video conferencing capability on NYNET and played an important role in demonstrating NYNET applications. However, the tool has limitations in handling a large number of participants and the maximum frame rate (for Video) that can be achieved. These limitations can be resolved by developing efficient techniques to perform group communication on ATM networks. These primitives will provide efficient multicasting, synchronization and management of all the participant processes involved in the conference. Furthermore, the current communication protocols (TCP/IP) do not efficiently support the communication services required in video conferencing. More research is needed to develop a communication protocol that efficiently provides the services required by video conferencing software.

3.3 Benchmarking ATM and Different Platforms

We experimented with the ATM API library and our results indicated that we were not getting good performance when compared with TCP/IP over ATM. For example, roundtrip time for 4096 bytes using TCP/IP over ATM is 4918 microseconds yielding a rough throughput (real bandwidth might be a little more) of 12.71 mbits/sec. Roundtrip time for 4096 bytes using ATM API over ATM (i.e. bypassing TCP/IP) is 4250 microseconds yielding a throughput of 14.71 Mbits/sec. This is much less than the 140 Mbits/sec bandwidth that can be provided by the switch.

In this experiment, we evaluated the communication latency between different computer architectures that are connected over ATM and/or Ethernet. Because of the wide use of the TCP/IP communication suite, our experimental results focus on benchmarking the performance of TCP/IP over the ATM network. IP packets are encapsulated in ATM PDUs using AAL 3/4 or 5 for segmentation, reassembling and framing of IP packets. Internet addresses are mapped to ATM 64-bit addresses using the ATM ARP protocol.

We have evaluated the communication latency between the following platforms.

- kepler - Sun IPX, SBA-200 ATM Sbus, SunOS 4.1.3
- hubble - Sun IPX, SBA-200 ATM Sbus, SunOS 4.1.3
- kopernik - SGI Challenge, VMA-200 ATM VMEbus, IRIX 5.2
- brahe - SGI Indy, GIA-100 ATM , IRIX 5.2
- newton - SGI Indigo, GIA-100 ATM, IRIX 5.2
- fore-atm - ASX-100 FORE switch

We have installed the 2.2.9 release of FORE software on all these platforms. We installed two Mosaic servers on SGI Challenge and Sun IPX and measured access and delivery time to a client via an ATM network and a dedicated Ethernet. Our benchmark results are summarized below:

- Mosaic client/server connection works faster for TCP/IP over ATM than over Ethernet, but the difference is not significant.
- IMAGE (gif file, 340Kbytes, res 1152x900)
 1. It takes 7 seconds to display the local gif file on SunIPX and 4 seconds on SGI Indy/Indigo using xv viewer in command line mode
 2. Mosaic takes about 8 seconds to download this file from the server, spawn xv viewer and display the file on Sun IPX
 3. We have noticed that Mosaic on SGI Indy/Indigo downloads gif files much slower when the ATM network is used than when dedicated Ethernet is used.
- The experiments with communication between SGI computers show that the TCP/IP over ATM is much SLOWER than TCP/IP over dedicated Ethernet. The problem lies in the default TCP window sizes defined in IRIX 5.2 UNIX kernel. The poor performance

of the ATM connection between SGI computers disappeared after modifying the TCP window sizes, socket space reservation and reconfiguration of UNIX kernels on all SGI workstations and SGI Challenge. Now the throughput for TCP/IP over ATM between SGI computers is around 20 Mbps and limited by end-stations.

- The waiting time for delivery of mpeg/jpeg/mvcl movie is almost unchanged when one starts three or four such connections simultaneously over ATM. It was much longer in the case of Ethernet. So, using ATM, we can increase the number of clients working simultaneously without any degradation in performance.
- The general problem with Mosaic images and movies is that Mosaic downloads the whole file from the server to the local disk and then spawns an external viewer/player. In the cases of ATM or dedicated Ethernet, the transfer time is determined by disk throughput on the client and server sides rather than network bandwidth. A 30 MByte movie file is downloaded by Mosaic in approximately 40 seconds (a throughput of 6Mbps) whereas the spawning of a movie player takes only 3 seconds.
- We tested the Fore Systems's user-level ATM library routines which provide an interface to the ATM data link layer. We checked a connection-oriented client/server model using SPANS signaling protocol and discovered that API over ATM is only slightly better than TCP/IP over ATM.
- The limitations which we observed in playing movies on SGI or Sun IPX are mainly due to the workstations (UNIX file system performance, frame buffer access and CPU speed).
- We haven't noticed any performance gain in digital video delivery when permanent virtual channels are used instead of switched virtual channels.
- FTP binary transfer of 27 MBytes takes 15 seconds over ATM and 31 seconds over Ethernet. It gives an average throughput (including disk I/O operations, access to memory, transfer, switch activity) of 14.4 Mbps (ATM) and 7.0 Mbps (dedicated Ethernet). These numbers vary slightly from computer to computer.
- The roundtrip of 4096 bytes measured by PING gives 3ms for ATM and 9ms for dedicated Ethernet.

Summary: We have evaluated the performance of the standard UNIX applications on both ATM and Ethernet. These applications include FTP, PING and CP as well as local previewers XV, MPEG-PLAY and MPEGMOVIE fired up manually or through Mosaic.

TCP/IP over ATM gives an average throughput of 16 Mbps (peak 21 Mbps) while TCP/IP over Ethernet gives an average throughput of 7 Mbps (peak 8.8 Mbps).

It takes about 21 seconds to make a copy of a local 27 MByte file on a local disk, 21 seconds to copy NFS mounted file via ATM and 38 seconds to copy NFS mounted file via Ethernet.

The performance of TCP/IP based applications running over ATM can be improved by tuning some kernel parameters (tcp_sendspace, tcp_recvspace, udp_sendspace, udp_recvspace)

The maximum theoretical speedups that can be obtained on the IPXs is around 49 Mbps on TCP and around 43 Mbps on UDP. The SGI should be able to give 80 Mbps.

3.4 Parallel/Distributed Software Tool Evaluation

In this task, we studied and evaluated the performance of different applications implemented using different tools and run on different platforms. We used three message passing tools Express, p4, and PVM for this benchmark. The computer architectures studied include IBM-SP1, Alpha cluster and SUN workstations. These computers are interconnected by one or more combinations of three networks; Ethernet, FDDI, ATM.

3.4.1 Primitives Supported by Different Software Tools

The primitives of any parallel/distributed software tool can be broadly characterized into four groups: Communication primitives, Synchronization primitives, Management/Control primitives and Exception Handling primitives.

The experimental results presented later evaluate the performance of send/receive, broadcast/multicast, ring communication and global summation primitives of the studied software tools (see Table 5).

These communication primitives play an important role in determining the performance of a large class of parallel/distributed applications. Hence, the tool that provides the best performance in executing its communication primitives will also give the best performance results for a large number of distributed applications.

3.4.2 Applications Benchmark Suite

Low level benchmark tests such as communication primitive performance can some time be misleading by suggesting performance advantages for one tool over another that may not

Table 5: Communication primitives for evaluating tools

Primitive	Express	p4	PVM
Send/Receive	exsend exreceive	p4_send p4_recv	pvm_send pvm_recv
Broadcast/Multicast	exbroadcast	p4_broadcast	pvm_mcast
Ring	exsend exreceive	p4_send p4_recv	pvm_send pvm_recv
Global Sum	excombine	p4_global_op	Not Available

Table 6: SU_PDABS

#	Numerical Algorithms	Signal/Image Processing	Simulation/Optimization	Utilities
1.	Fast Fourier Transform	JPEG Compression	N-body Simulation	ADA Compiler
2.	LU Decomposition	Hough Transform	Monte Carlo Integration	Parallel Sorting
3.	Linear Equation Solver	Ray Tracing	Traveling Salesman	Parallel Search
4.	Matrix Multiplication	Data Compression	Branch and Bound	Distributed Spell Checker
5.		Cryptology		Distributed Make

be relevant in actual applications. So in this level, we evaluate the tools from the application performance perspective. We have used different classes of applications from the parallel/distributed applications benchmark suit (SU_PDABS) that is currently being developed at NPAC (Northeast Parallel Architectures Center) at Syracuse University.

We have divided the applications into four classes: Numerical algorithms, Signal/Image Processing applications, Simulation/Optimization applications, and Utilities. Applications under different classes are shown in Table 6. We have chosen applications to include simple, medium, and complex problems to represent a broad spectrum of applications. Even though it covers a broad spectrum of applications, it is not comprehensive. All applications in this suite are written in C using different distributed/parallel tools (i.e. Express, p4, and PVM).

From this benchmark suite, we have chosen JPEG Compression, Fast Fourier Transform (FFT), Monte Carlo Integration and Parallel Sorting applications for benchmarking the software tools.

3.4.3 Experimental Results

In this subsection, we discuss the experimental results of the tool primitives and performance of the applications when implemented on different platforms using different tools. These results can be used to assist in determining the best platform, network technology, and software tool to run a given class of applications.

1. **Software Tool Primitives' Results:** In what follows, we benchmark the point-to-point and group communication primitives of three parallel/distributed software tools on different distributed computing platforms.

(a) **Send/Receive Primitives:** Table 7 shows the execution time of send/recv primitives when implemented in Express, p4 and PVM and for different message sizes up to 64 Kbytes. For example, for a message size of 16 Kbytes, the send/recv primitive takes approximately 111, 44, and 61 milliseconds when it is implemented using Express, p4 and PVM respectively over Ethernet. It is clear from this table that the p4 implementation of point-to-point communications on SUN Workstations has the best performance when compared to the other tool implementations.

Table 7 shows the send/recv time for these tools on SUN SPARCstations over ATM LAN and NYNET. Similarly to the Ethernet results, the p4 implementation of the send/receive primitives outperformed the other tool implementations. Express performs a little better than PVM for small message sizes (up to 1 Kbyte), but PVM outperforms Express for large messages. This table shows the significant improvement in throughput when ATM networks are used as the underlying communication network of high performance distributed systems. Furthermore, this table shows that NYNET performance of send/receive primitives is similar to those of ATM LAN. Hence, it is feasible to build distributed computing systems across the NYNET and their performance should be comparable to those based on LANs.

- (b) **Broadcast Primitives:** For this group communication primitive, p4 has the best performance while Express has the worst performance. It is worth noting that the tool with better send/recv performance does not necessarily imply the better performance for broadcast/multicast primitives. This is because of the fact that broadcast/multicast performance greatly depends on the algorithm used for its implementation. We observed similar results on the NYNET network.
- (c) **Ring Communication:** Ring communication was implemented using send/recv primitives in all three tools. As with other communication primitives, p4 performed

Table 7: send/recv timing for SUN SPARCstations (in milliseconds)

Mesg Size (Kbytes)	PVM			p4			Express	
	Ethernet	ATM (LAN)	NYNET	Ethernet	ATM (LAN)	NYNET	Ethernet	ATM (LAN)
0	9.655	7.991	7.764	3.199	2.966	3.636	4.807	4.152
1	11.693	8.678	8.878	3.599	3.393	4.168	10.375	7.240
2	14.306	9.896	10.105	4.399	3.748	4.822	18.362	11.061
4	25.537	13.673	14.665	9.332	4.404	5.069	32.669	16.990
8	44.392	18.574	19.526	24.165	6.482	7.459	59.166	27.047
16	61.096	27.365	28.679	44.164	11.191	13.573	111.411	46.003
32	109.844	48.028	53.320	98.996	19.104	22.254	189.760	82.566
64	189.120	88.176	91.353	173.158	35.899	41.725	311.700	153.970

best among all other tools. One interesting point to note is that even though PVM performed better than Express in the send/recv primitive, Express outperformed PVM for ring communication. This indicates that Express is better suited for continuous flow of incoming and outgoing data when compared to PVM. However, p4 is the best among the three for this type of applications.

- (d) **Global Summation:** Global operations are very important in measuring performance of PDC tools. We selected global summation for our performance measurement as this is the most commonly used global operation. PVM does not support any global operation and thus it is not evaluated for this operation. For this global operation, the p4 implementation is also better than Express.

Table 8 summarizes the results of our evaluation of these tools with respect to their communication primitives. From this table, we can see that p4 outperformed Express and PVM in all classes of communication primitives. This can be attributed to the efficient implementation of p4 communication primitives which add very small amounts of overhead to the underlying transport layer.

2. **Applications' Performance:** We evaluated the parallel/distributed software tools by comparing the execution times of four applications (JPEG Compression, Two-Dimensional Fast Fourier Transform, Monte Carlo Integration, Sorting by Regular Sampling) that are commonly used in distributed systems.

We have benchmarked these applications on all the platforms discussed before and when they are implemented using p4, PVM, and Express tools.

Table 8: Summary of Tool Performance on Different Platforms

SUN/Ethernet				SUN/ATM		
send/recv	broadcast	ring	global sum	send/recv	broadcast	ring
p4	p4	p4	p4	p4	p4	p4
PVM	PVM	Express	Express	PVM	PVM	PVM
Express	Express	PVM		Express		

For the ALPHA cluster, the p4 implementation of JPEG compression and 2D-FFT performed the best whereas PVM and Express implementations were best for sorting and Monte Carlo integration, respectively. Since JPEG compression involves heavy communication, the p4 implementation of JPEG compression understandably performs the best, since it involves the least communication overhead among all three tools as shown in the previous subsection.

For IBM-SP1, the results are consistent with those obtained on the ALPHA cluster. However, the execution times are significantly higher on IBM-SP1 compared to the ALPHA cluster, because SP1 uses slower processing nodes and interconnect network.

Comparing the applications' performance when they are implemented on NYNET (ATM WAN) and on Ethernet LAN shows that distributed computing is feasible across wide area networks and can outperform LANs if higher speed network technology such as ATM is used.

3.4.4 Discussion

Although many criteria have been excluded while evaluating the software tools, the results presented above show where the tools stand as far as performance is concerned. Many details like application development (how easy is it to develop an application using a given tool), capabilities of a tool to support debugging and user interface should be taken into consideration when necessary.

4 Demonstrations

We have demonstrated the capabilities of NYNET to provide the communication, storage and computations for large scale HPCC applications. In what follows, we briefly describe the main NYNET demonstrations that were organized to show the benefits that can be gained from running the HPCC applications developed in this project over NYNET.

Congress Demo: This demonstration was given to the U.S. House Representatives Subcommittee on Science, Space and Technology on October 25, 1993. The applications which were demonstrated included Concurrent Multitarget Tracking System, Financial Modeling application, Electromagnetic Scattering Simulation (which have been discussed in the previous sections of this report) and Integrated Multimedia Environment which is briefly described below.

Integrated Multimedia Environment demonstrated the use of a high speed network and multimedia technology to reduce health costs and to improve the quality of providing health care. It also demonstrated the capability to enhance education and support military operations. This demo used commercial multimedia software, Communique!, developed by InSoft Inc., that allows the transfer of audio, video, text, and images over a TCP/IP network. The description of the software is given in one of the previous sections. This technology is still in its infancy and Syracuse University researchers are working on improving this technology by developing multimedia communications software that utilizes the high bandwidth offered by NYNET. This application demonstrated the following functions:

- Audio/Video Teleconferencing
- Transfer of medical images and how physicians interconnected by a high speed network can collaborate on studying patient images. Currently, regular/express mail and phone conferencing are used to achieve this task. In this function, we transferred a hand x-ray image and an ultra sound image.
- Education. A delicate surgery (e.g., open heart surgery) can be transferred to medical students across high speed networks. Several specialists can collaborate on performing a medical procedure performed by another doctor at a remote location (e.g, disadvantaged place). Here, we showed a medical operation performed on an arm by using a VCR tape provided by Upstate hospital.
- Military. Two commanders can discuss a battle scenario by using detailed diagrams related to the theater of operation. In this scenario, the image of Griffiss Air Force Base is transferred and discussed. This application demonstrated the use of high speed networking and multimedia technology in medicine, education, and the military.

Visit of Hillary Clinton: On April 5, 1994, First Lady Hillary Clinton and Senator Daniel Patrick Moynihan visited NPAC to witness how the InfoMall is helping to integrate today's promising high performance computing and communications (HPCC) technologies for important applications in industry. Prof. Geoffrey Fox has demonstrated an experimental

telemedicine system running over NYNET. Using this telemedicine system, it was demonstrated how doctors could use this technology to analyze multimedia information on patients at remote locations. Ms. Clinton watched as doctors in NPAC and doctors in Rome Labs were communicating with each other through audio and video about a child's condition.

NYNEX/Rome Labs Demo: Several High Performance Communication and Computation (HPCC) applications were used to demonstrate the capabilities of NYNET to members of NYNEX. This demonstration took place at NPAC. The applications included multimedia language learning software developed by Syracuse Language Systems. This software uses audio and pictures for teaching a new language. The setup included two PCs and a UNIX server connected by Ethernet. The software and data are located in the server and are accessed from PCs. The idea of Education on Demand (PCs in class rooms accessing data in remote servers via high speed networks) was demonstrated here. Other applications included a Mosaic server and Video on Demand applications.

TOA/COA Conference Demonstration This demonstration was a part of the conference held at the Sheraton Inn, Syracuse, NY during June 6, 7, and 8, 1994. NYNEX has added a temporary off ramp to NYNET. Prof. Geoffrey Fox has demonstrated the Video on Demand applications which use the high bandwidths of ATM NYNET and computing capabilities available at NPAC. During these demonstrations, we have experienced network routing problems that have prevented us from accessing the parallel computers at NPAC. This problem was caused by IP traffic routing tables. This problem triggered the need to have a uniform IP routing technique to be used by all NYNET participants.

5 Summary and Conclusions

In this project, we have developed in a relatively short period of time, a wide range of HPCC applications and demonstrated the benefits of running these applications on a high speed wide area network such as the NYNET. The applications include Multitarget Tracker, Financial Modeling application, Electromagnetic Scattering, JPEG compression, and Fractal generation. We also evaluated the use of Mosaic as a user interface to launch HPCC applications running on geographically dispersed high performance computers ranging from supercomputers or parallel computers down to desktop computers.

This project has identified several limitations in current HPCC technologies that must be addressed. These areas and research issues are summarized below:

1. Need for an efficient communication system: TCP/IP protocols were designed in the days when bandwidth was not high and there were frequent errors in transmission. As a result, TCP/IP protocols add a lot of overhead to provide error detection, flow control, etc., and are not suitable for high speed ATM networks like NYNET. Thus, there is a need for an efficient communications system which is suitable for high speed networks. More research is needed to make communication protocols efficient.
2. Multimedia software: Multimedia is still in its infancy and more research is needed to develop an evaluation methodology and to improve their performance. The proliferation of powerful personal computers will play an important role in the widespread use of PCs to access and run multimedia applications across high speed networks. The use of PCs as client machines to access multimedia servers raises interesting issues that need to be investigated. The performance of multimedia applications when the PCs access the NYNET through Ethernet or the ISDN network should be studied. Currently, existing multimedia software does not support efficient collaboration among a large number of participants. More research is needed on how to improve their performance and scale their capability so that a large number of users interconnected over NYNET can collaborate and interact to solve large scale applications. The goal of one of the projects at NPAC is to develop efficient techniques for delivering multimedia applications over the NYNET through the ISDN network.
3. Information services on Demand: The use of high speed networks and high performance computers will facilitate the deployment of information servers that can be accessed over a high speed network like NYNET. Questions on how the server's information should be accessed needs further research. Two possibilities for this are 1) a local server acts as a cache and provides required services to the clients and 2) use several remote servers that can be accessed concurrently by the clients through the NYNET. One important class of such information servers is the Video On Demand (VOD) server. More research is needed to evaluate the best platform (parallel/distributed) and to develop and implement VOD on NYNET. Also, what type of communication protocols are best suited for VOD applications should be addressed.
4. ISDN and B-ISDN internetworking: The ISDN is intended to be a worldwide public telecommunication network to replace existing public telecommunication networks and deliver a wide variety of services. One important problem that must be addressed is the internetworking of ISDN and ATM based B-ISDN. Proliferation of the use of information servers will grow explosively when the issue of internetworking of ISDN and

B-ISDN networks is resolved. By solving this issue, we can allow 100 or 1000 PCs to access NYNET information servers by using the ISDN network. Access then is as easy as dialing the number of the required information server. This allows the users to access NYNET servers from anywhere in the world.

References

- [1] T. D. Gottschalk, "CALTRAX The Tracking Program for Simulation 87", *Caltech Report C3P-478*, California Institute of Technology, Pasadena, California 91125.
- [2] Salim Hariri et. al. "Parallel Software Benchmark for BMC³/IS Systems", Northeast Parallel Architectures Center, 111 College Place, Syracuse University, Syracuse, NY.
- [3] F. Black, and M. Scholes. "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy*, 81, 1973, 637-59. 1973.
- [4] T. Finucane, "Binomial Approximations of American Call Option Prices with Stochastic Volatilities," published in *Journal of Finance*. 1992.
- [5] Mills, K., Vinson, M. and Cheng, G., "A Large Scale Comparison of Option Pricing Models with Historical Market Data," in *Proc. of the 4th Symposium on the Frontiers of Massively Parallel Computation*, McLean, VA, IEEE Computer Society Press, October 1992.
- [6] G. Cheng, K. Mills and G. Fox, "An Interactive Visualization Environment for Financial Modeling on Heterogeneous Computing Systems," *Proc. of the 6th SIAM Conference on Parallel Processing for Scientific Computing*, R. F. Sincovec, eds., SIAM, Norfolk, VA, March 1993.
- [7] Mills, K., Cheng, G., Vinson, M., Ranka, S. and Fox, G., "Software Issues and Performance of a Stock Option Pricing Model on the Connection Machine-2 and DECmpp-12000," in *Proc. of Fifth Australian Supercomputing Conference*, Melbourne, Australia, December, 1992.
- [8] G. Cheng, G. Fox, K. Mills and Marek Podgorny, "Developing Interactive PVM-based Parallel Programs on Distributed Computing Systems within AVS Framework," to be presented at the 3rd Annual International AVS Conference, JOIN THE REVOLUTION: AVS'94, Boston, MA, May 2-4.

- [9] Y. Lu, A. G. Mohamed, G. Fox and R. F. Harrington, "Implementation of Electromagnetic Scattering from Conductors Containing Loaded Slots on the Connection Machine CM-2", in Proc. of the 6th SIAM Conference on Parallel Processing for Scientific Computing, March 1993, Norfolk, VA.
- [10] G. Cheng, Y. Lu, G. C. Fox, K. Mills and T. Haupt, "An Interactive Remote Visualization Environment for an Electromagnetic Scattering Simulation on a High Performance Computing System," in the Proceedings of Supercomputing '93, Portland, Oregon, November 15-19, 1993.

MISSION
OF
ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.